# Lecture 6

```
#include <iostream>
#include <ostream>
using namespace std;

class matrix {
  int row, col;
  float **mat;

public:
  matrix(int r = 1, int c = 1) {
    row = r;
    col = c;
    mat = new float *[row];
    for (int i = 0; i < row; i++) {
      mat[i] = new float[col];
    }
  }
  // add two matrix
  matrix operator+(matrix &m) {
    matrix temp(row, col);
    for (int i = 0; i < row; i++) {
      for (int j = 0; j < col; j++) {
        temp.mat[i][j] = mat[i][j] + m.mat[i][j];
      }
    }
    return temp;
  }
  // subtract two matrix
  matrix operator-(matrix &m) {
    matrix temp(row, col);
    for (int i = 0; i < row; i++) {
      for (int j = 0; j < col; j++) {
        temp.mat[i][j] = mat[i][j] - m.mat[i][j];
      }
    }
    return temp;
  }
  // multiply two matrix
  matrix operator*(matrix &m) {
    matrix temp(row, col);
    for (int i = 0; i < row; i++) {
      for (int j = 0; j < col; j++) {
        temp.mat[i][j] = mat[i][j] * m.mat[i][j];
      }
    }
    return temp;
  }
  // display matrix
  void display() {
    for (int i = 0; i < row; i++) {
      for (int j = 0; j < col; j++) {
```

```
        cout << mat[i][j] << " ";
      }
      cout << endl;
    }
  }
  // add one to each matrix element
  matrix operator++() {
    for (int i = 0; i < row; i++) {
      for (int j = 0; j < col; j++) {
        mat[i][j] = mat[i][j] + 1;
      }
    }
    return *this;
  }
  // subtract one from each matrix element
  matrix operator--() {
    for (int i = 0; i < row; i++) {
      for (int j = 0; j < col; j++) {
        mat[i][j] = mat[i][j] - 1;
      }
    }
    return *this;
  }
  //  +=
  matrix operator+=(matrix &m) {
    for (int i = 0; i < row; i++) {
      for (int j = 0; j < col; j++) {
        mat[i][j] = mat[i][j] + m.mat[i][j];
      }
    }
    return *this;
  }
  //  -=
  matrix operator-=(matrix &m) {
    for (int i = 0; i < row; i++) {
      for (int j = 0; j < col; j++) {
        mat[i][j] = mat[i][j] - m.mat[i][j];
      }
    }
    return *this;
  }
  //   get element
  float get(int r, int c) { return mat[r][c]; }
  // set element
  void set(int r, int c, float val) { mat[r][c] = val; }
  //   transpose matrix
  matrix transpose() {
    matrix temp(col, row);
    for (int i = 0; i < row; i++) {
      for (int j = 0; j < col; j++) {
        temp.mat[j][i] = mat[i][j];
      }
    }
    return temp;
  }
};

int main() {
```

```
  matrix mymat1(2, 2);
  mymat1.set(0, 0, 1);
  mymat1.set(0, 1, 2);
  mymat1.set(1, 0, 3);
  mymat1.set(1, 1, 4);
  matrix mymat2(2, 2);
  mymat2.set(0, 0, 1);
  mymat2.set(0, 1, 2);
  mymat2.set(1, 0, 3);
  mymat2.set(1, 1, 4);
  matrix mymat3(2, 2);
  mymat3 = mymat1 + mymat2;
  cout << mymat3.get(0, 0) << endl;
  cout << mymat3.get(0, 1) << endl;
  cout << mymat3.get(1, 0) << endl;
  cout << mymat3.get(1, 1) << endl;
  mymat1.display();
  cout << endl;
  mymat2.display();
  cout << endl;
  mymat3.display();
  matrix mymat4(2, 2);
  mymat4 = mymat1.transpose();
  cout << endl;
  mymat4.display();
  cout << endl;
  mymat4 += mymat1;
  mymat4.display();
  return 0;
}
```